# Methodological notes

P7 Data Scientist OpenClassrooms

Started on Monday the 28th of November 2022

Oumeima EL GHARBI

# Training method

## Dataset

This project is about predicting if a client of the bank will be able to repay his loan.

We have a training dataset containing 307551 observations / applications for a loan.

Of these 307551 applications, 92% belong to the class 0 which means that 92% of the clients repaid their loan.

## Feature Engineering

Based on the following kernel : https://www.kaggle.com/code/jsaguiar/lightgbm-with-simple-features/script

Models trained :

- Logistic Regression (standardisation)
- Random Forest
- HistBoost
- LightGBM

## Hyper-parameters tuning :

- Logistic Regression
- Random Forest
- HistBoost

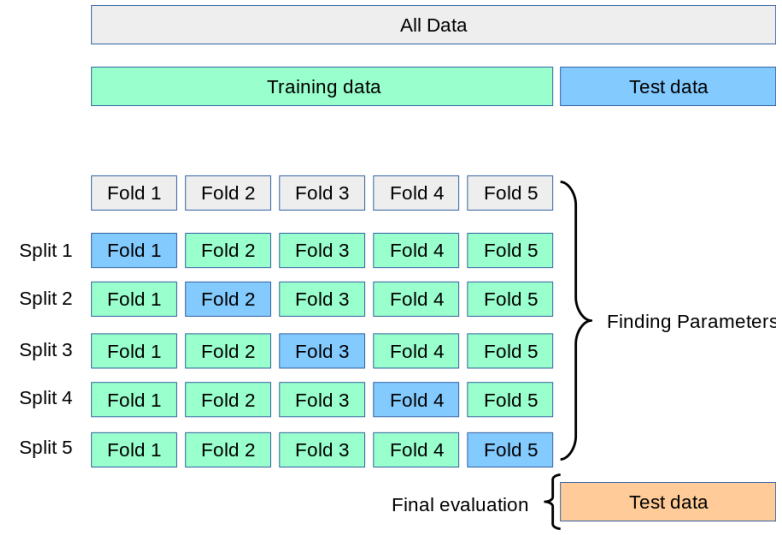
For LightGBM, due to performance and time constraints, we have used the hyper-parameters from the Feature Engineering kernel.

## Cross Validation

We have used Stratified K folds

- K = 5 :
- Logistic Regression
- Random Forest
- HistBoost

- K = 10 for the LightGBM




The training method of the LightGBM is a bit different from the other models' since we have kept missing values and infinite values.
This was so that we could model as realistically as possible the data.

## SMOTE

"Over sampling is used when the amount of data collected is insufficient. A popular over sampling technique is SMOTE (Synthetic Minority Over-sampling Technique)" which duplicates observations from the minority class to balance the dataset.

We have tried SMOTE oversampling on :

- Logistic Regression
- Random Forest
- HistBoost

However, since the results weren't that helpful, we didn't use it when training the LightGBM.

## Training

- We have separate, using train_test_split from sklearn, the preprocessed dataset from the feature engineering into two sets.
- One dataset is for training and is called train.csv and contains 70% of the observations.
- The other dataset was made for testing purposes. It contains 30% of the data and will be used to evaluate and compare the different classification algorithms.
- For the training, we generate K Folds, then we define X_train (K – 1 folds) and X_val (1 fold). That method enables us to have a more robust model and more trust in our predictions.

# Metrics and Cost function

## Evaluation metrics

### Use case metrics

| | | Predicted | |
|---|---|---|---|
| | | **Negative** | **Positive** |
| **Actual** | **Negative** | True Negative | False Positive |
| | **Positive** | False Negative | True Positive |

- In our case, predicting 0 (Negative) means that the client will repay the loan.
- Predicting 1 (Positive) means that the client will not repay the loan (**default risk**).
- We have an imbalanced dataset : our database of clients for which we know if they have repaid their loan contains 92% of clients that have repaid it while 8% had a default of payment.

Accuracy : not trustful because the dataset is imbalanced : Dummy predict only 0 and gets an accuracy of 92%

### ROC Curve and AUC

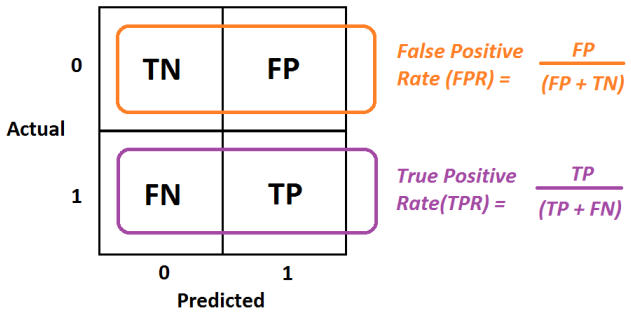
For the Kaggle competition, AUC-ROC score was chosen to compare the different model performances.

**True Positive Rate** (**TPR**) is a synonym for recall and is therefore defined as follows:

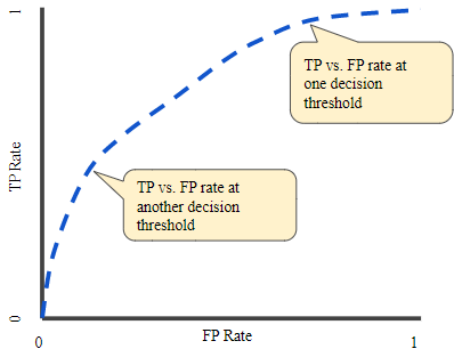
$$TPR = \frac{TP}{TP + FN}$$

**False Positive Rate** (**FPR**) is defined as follows:

$$FPR = \frac{FP}{FP + TN}$$

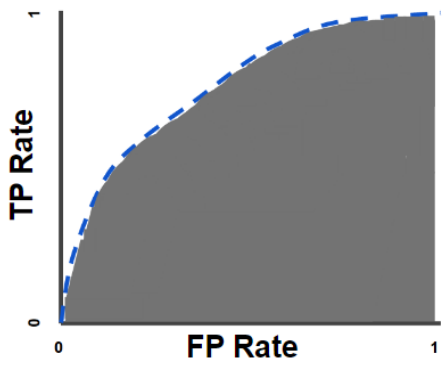



#### *ROC curve*




Aaa

*AUC: Area Under the ROC Curve*




Google : “AUC stands for "Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus) from (0,0) to (1,1).”

https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

## Precision

$$\text{Precision} = \frac{TP}{TP + FP}$$

| | | Predicted | |
|---|---|---|---|
| | | **Negative** | **Positive** |
| **Actual** | **Negative** | True Negative | False Positive |
| | **Positive** | False Negative | True Positive |

- In our case :
- FP (False Positive) means that our model predicted 1 when it should be 0.
- If we have False Positive, it means that our model predicted that the client will not repay the loan when in fact the client might repay it.
- This means the bank (Home Credit) lost one potential client that could have repaid the loan without trouble.
- Thus, if we have a high precision, then we won’t miss potential client that could have repaid the loan. A low precision means we will miss “good” clients.
- However, the banker can decide to give a loan by studying the client’s application more in detail.

## Recall

$$\text{Recall} = \frac{TP}{TP + FN}$$

| | | Predicted | |
|---|---|---|---|
| | | **Negative** | **Positive** |
| **Actual** | **Negative** | True Negative | False Positive |
| | **Positive** | False Negative | True Positive |

- FN (False Negative) means that our model predicted 0 while it is positive.
- **This means that our model predicted that the client will repay the loan while in fact the client will not repay it .**
- This is important for our bank. Indeed, we do not want to have a high number of False Negative.
- Having a high number of False Negative means that Home Credit gave a loan to clients that might not repay it !
- Therefore, we want to reduce as much as possible giving loans to clients that might not pay it back, thus, we need to lower the number of False Negative which means we want to get a good recall.

*Conclusion* : precision / **recall** : we prefer having a good recall instead of a good precision. This means that we want to lower the number of False Negative which means that we do not want to give a loan to clients that might not pay it back.

## Loss / Cost function

### F-score

*F1-score (beta = 1)* : recall as important as precision

$$F_1 = \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = 2\frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{\mathrm{tp}}{\mathrm{tp} + \frac{1}{2}(\mathrm{fp} + \mathrm{fn})}.$$

*F Beta score*

A more general F score, FB, that uses a positive real factor B, where B is chosen such that recall is considered B times as important as precision :

$$F_\beta = (1 + \beta^2) \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{(\beta^2 \cdot \mathrm{precision}) + \mathrm{recall}}.$$

We have to choose a value for Beta with B > 0.

- **As for the F-score, we need to give more importance to the recall for our use case.**

For example, Beta = 2 means that recall is twice more important than precision.

*From sklearn documentation :*

*“The beta parameter determines the weight of recall in the combined score. beta < 1 lends more weight to precision, while beta > 1 favors recall.*

*For the limits, when beta -> 0, we consider only precision,*

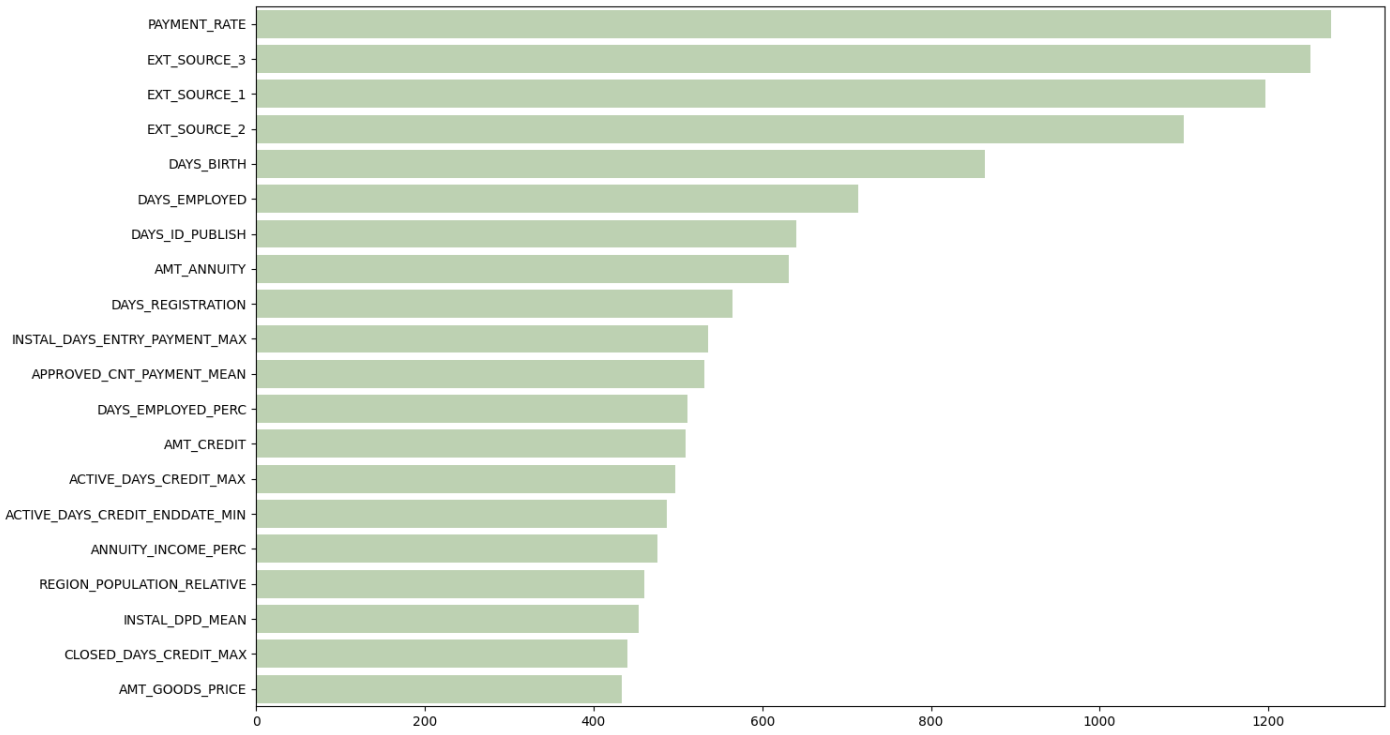
*If beta -> +inf only recall).”*

**Conclusion** : in our use case, we need a Beta > 1, we chose beta = 2

# Global and Local Interpretability

## Global : feature importance

The global feature importance was computed using the model's attribute. We give each feature a value of importance. That number represents how important the feature is for the model make a decision (based on the logic of decision trees).

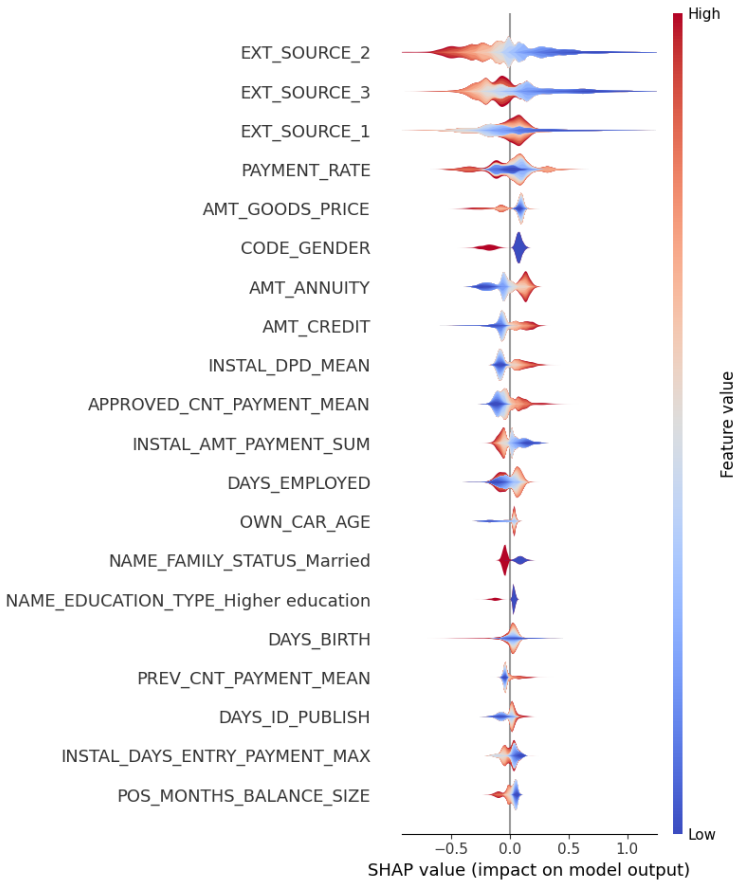



## Local : SHAP values (SHapley Additive exPlanations)

The local feature importance, like the global feature importance, determines which features are the most important.

Shapley values can be used to assess local feature importance. They are used to explain which features contribute most to a specific prediction.

Shapley values are different for each observation's prediction.



## Limits and improvements to make

### Data

- We can improve the data cleaning and feature engineering
- We could also use feature selection so that we get a better performance of the model and less computing time (3 hours for the LightGBM)
- Due to time constraints, we didn't get a complete understanding of all the features
- We could work on a deeper exploratory data analysis

### Prediction models

- Try other classification models (more complex models like Neural Network)
- Improve the hyper-parameters tuning
- Optimize the training time
- Get better scores
- Track the experiments using MLFlow for example

### Imbalanced data

- Try other methods to balance the dataset (under sampling for instance)

### API

- Add validation of parameters using pydantics
- Make the code more robust to time and changes by adding tests (unittest / pytest)